VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multiplatformní vývoj mobilních aplikací založený na Java technologii

Multiplatform Development of Mobile Application Based on Java Technology

2020 Martin Halfar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Halfar**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Multiplatformní vývoj mobilních aplikací založený na Java technologii
Multiplatform Development of Mobile Application Based on Java Technology

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je výzkum možností multiplatformního vývoje pro platformy Android a iOS prostřednictvím technologie Java. Součástí práce bude také vytvoření ukázkové aplikace, která demonstruje možnosti vybraného nástroje.

Práce bude zejména obsahovat:
1. State of the Art dostupných nástrojů pro multiplatformní vývoj mobilních aplikací.
2. Výběr a zdůvodnění nástroje, který využívá Java technologii.
2. Návrh demonstrační aplikace.
3. Implementaci demonstrační aplikace.
4. Srovnání vybraného nástroje s jinými z pohledu funkcionality i výkonu. Součástí budou experimenty zaměřené na výkon.

Seznam doporučené odborné literatury:

[1] BENNETT, Jim, 2018. Xamarin in Action: Creating native cross-platform mobile apps. 1 edition. Shelter Island: Manning Publications. ISBN 978-1-61729-438-9.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kožusznik, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2020 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2020 .....

Rád bych na tomto místě poděkoval všem, kteří se podíleli na mých ostatních projektech. Pouze díky tomu mohla tato práce vzniknout.

Abstrakt

Účelem práce je shrnutí několika nástrojů pro multiplatformní vývoj aplikací, zejména pak pro platformy Android a iOS. Práce obsahuje stručný popis každého z vybraných nástrojů a poukazuje na jeho výhody či nedostatky. Nástroje byly vybrány tak, aby byla v zastoupení co nejširší řada programovacích jazyků. Každý nástroj je pak otestován z hlediska výkonu, kdy závěr práce obsahuje tabulky s výsledky testů. Kromě samotných testů také práce obsahuje mobilní aplikaci vytvořenou v jazyce Java pomocí nástroje Gluon. Tento nástroj byl vybrán výhradně z toho důvodu, že se jedná o jediný nástroj založený na technologii Java a jedná se o nástroj dostatečně kvalitní. V případě, že by neplatilo omezení na technologii Java, byl by vybrán nástroj Adobe PhoneGap nebo jiný, založený na webových technologiích (JavaScript, případně TypeScript).

Klíčová slova: multiplatformní vývoj, Gluon, Java, Android, iOS

Abstract

The goal of this thesis is to summarize few of available tools for multiplatform development of mobile applications, mainly for the platforms Android and iOS. The thesis contains short descriptions for each of selected tools and describes its advantages and disadvantages. The tools were selected in a way to show as many programming languages as possible. Every tools is tested for its performance, where the conclusion includes tables with the test results. Other than the tests, this thesis also contains mobile application created in Java with the use of the tool Gluon. The reason behind selection of this tool was nothing other than it being the only Java-based tool while being of useable quality. If there were no language-limitations, then the Adobe PhoneGap or other web-based tool would be used instead (JavaScript or TypeScript).

Keywords: multiplatform development, Gluon, Java, Android, iOS

Obsah

Seznam použitých zkratek a symbolů

SDK	–	Software Development Kit
API	–	Application Programming Interface
CLI	–	Command Line Interface
CSS	–	Cascading Style Sheets
TS	–	TypeScript
JS	–	JavaScript
HTML	–	HyperText Markup Language
QML	–	Qt Modeling Language
JSON	–	JavaScript Object Notation
GPS	–	Global Positioning System
GUI	–	Graphical User Interface
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

1 Pojmy

1.1 Mobilní zařízení

Mobilní zařízení je pojem, který označuje skupinu přenosných počítačů. Ty obvykle obsahují integrovaný display, tlačítkové či dotykové ovládání, integrovanou baterii a širokou řadu konektivity (Wi-Fi, GPS, Bluetooth a další). V praxi se jedná převážně o chytré telefony či tablety s vlastním operačním systémem.

1.2 Mobilní aplikace

Mobilní aplikace je program, který lze na mobilní zařízení nainstalovat a spustit. Účelem takovéto aplikace je rozšíření funkcionality zařízení o funkce, které mobilní zařízení obvykle neobsahuje. Většina mobilních zařízení dnes obsahuje řadu aplikací, které jsou zde nainstalovány přímo od výrobce.

1.3 Fragmentace

Fragmentace [1] je pojem označující rozdělení mobilních zařízení podle platformy, systému či hardwarové konfigurace. Jedná se o problém, který se snaží nástroje pro multiplatformní vývoj aplikací částečně vyřešit.

1.4 Vývojové prostředí

Vývojové prostředí je sada nástrojů, kterých je třeba k vývoji, testování a sestavení produktu. Základním prvkem každého vývojového prostředí je textový editor.

1.5 SDK

SDK je sada nástrojů, které je nutné použít v případě, že chceme vytvořit mobilní aplikaci pro danou platformu. Každá platforma má své vlastní SDK a liší se i dle verze.

1.6 CLI

CLI je program, které využívá ke své funkci konzoli místo grafického uživatelského rozhraní. Většinou se jedná o programy, které jsou mnohdy funkčně rozsáhlejší než jejich GUI alternativy.

1.7 Multiplatformní vývoj

Multiplatformní vývoj znamená, že vývoj aplikace probíhá pouze jednou a výsledkem bude vlastní aplikace pro každou z vybraných platforem.

1.8 Nativní aplikace

Nativní aplikace je mobilní aplikace vytvořená pomocí nástroje, který dodává tvůrce dané platformy. Aplikace je schopná fungovat pouze na své platformě a s ostatními platformami není kompatibilní. Tento typ aplikací se vyznačuje zejména svou rychlostí, kvalitou a věrností uživatelského rozhraní a téměř neomezeným přístupem k funkcím mobilního zařízení. Tyto aplikace mohou vždy využít těch nejnovějších prvků. K jejich vývoji také existuje obsáhlá dokumentace a rozsáhlá řada návodů, které jsou dodávány tvůrcem dané platformy.

1.9 Webové aplikace

Webové aplikace jsou velmi podobné klasickým webovým stránkám, kdy ke své funkci využívají webový prohlížeč na cílovém mobilním zařízení. Aplikaci není nutné nijak instalovat ani aktualizovat, protože se její obsah pokaždé stáhne do zařízení. Díky tomu je ale aplikace oproti nativní aplikaci horší z hlediska výkonu i šíře funkcionality, ke které má aplikace přístup. Taková aplikace není spustitelná, pokud není přítomno připojení k internetu.

Schéma webové aplikace je identické s webovými stránkami, kdy jsou jednotlivé stránky vytvořené pomocí HTML a CSS, a jako logika aplikace slouží skript psaný pomocí JavaScriptu, či některým z jeho alternativ.

1.10 Hybridní aplikace

Hybridní aplikace je spojení webové a nativní aplikace. Taková aplikace je většinou rozdělená na dvě části, kdy vývojář pracuje pouze s webovou částí a nativní část je dodána při sestavení aplikace. Nativní část zde slouží jako most mezi webovou částí a zařízením a poskytuje volný přístup k funkcionalitě zařízení, stejně jako u nativních aplikací. Webová část je zde umístěna ve webovém kontejneru místo webového prohlížeče.

Aplikaci je nutné nainstalovat a udržovat aktualizovanou, protože všechna data jsou uložena v cílovém zařízení. Zde není potřeba, aby bylo mobilní zařízení připojeno k internetu.

1.11 Interpretované aplikace

Interpretované aplikace jsou aplikace, u kterých dochází ke generování kódu tak, aby využívaly originální prvky uživatelského rozhraní jako u nativních aplikací. Logika je zde tvořena pomocí jazyků Java, Ruby nebo XML. Výhodou takovéto aplikace je to, že se chová stejně jako nativní aplikace. Nevýhodou interpretovaných aplikací je závislost na daném vývojovém prostředí, kdy prostředí musí všechny prvky uživatelského rozhraní podporovat.

1.12 Generované aplikace

U generovaných aplikací dochází k vygenerování více aplikací z jednoho projektu, kdy se pro každou platformu vytvoří vlastní aplikace. Toto se stane až při sestavení aplikace.

Tento způsob tvorby mobilních aplikací pro vícero platforem je velmi efektivní, protože výsledná aplikace je v podstatě identická s nativní aplikací z pohledu výkonu a přístupu k funkcím zařízení. Problém u těchto aplikací je nemožnost kompletní optimalizace aplikace, protože výsledný kód závisí na použitém generátoru, který není možné jednoduše ovlivnit.

1.13 Překladač

Překladač je nástroj, který se stará o převod kódu mezi dvěma programovacími jazyky. Obvykle mezi jazyky, které jsou mezi sebou syntakticky podobné, například Java a CSharp. Tento nástroj je možné použít i při multiplatformním vývoji aplikací, ale samotný k vývoji nestačí.

2 Úvod

Vývoj aplikací pro mobilní zařízení je dnes mnohem náročnější, než je tomu u klasických desktopových aplikací. Důvodem proč je tomu tak, je unikátnost každého mobilního zařízení.

Každý výrobce mobilních zařízení má k dispozici různé platformy a hardware. Kromě toho se můžou dvě zařízení, které jsou shodné z pohledu konfigurace hardwaru, lišit verzemi operačního systému.

Většina služeb má dnes vlastní mobilní aplikaci, u které se dnes očekává to, že bude existovat na všech populárních mobilních platformách (zejména se jedná o Android a iOS). Bez existence speciálních nástrojů, které nám umožní multiplatformní vývoj aplikací, je jedinou možností použít nástroje poskytované výrobci platforem a pro každou platformu vyvinout aplikaci zvlášť.

Taková aplikace se označuje jako nativní. Nativní aplikace mají ten nejširší přístup k funkcím zařízení. K těmto patří například kamera, geolokace, akcelerometr, notifikační systém a další. Výkon nativní aplikace závisí pouze na hardwarové konfiguraci cílového zařízení. Nevýhodou je zde nemožnost využít zdrojový kód znovu během vývoje aplikace pro jiné platformy. To v praxi znamená, že práce nutná k vývoji nebo aktualizaci aplikace se násobí počtem platforem, na které aplikaci potřebujeme.

2.1 Možnosti vývoje pro více platforem

Vývoj aplikace pro vícero platforem je možné řešit několika způsoby. Je ale nutné podotknout, že jednodušší nebo levnější vývoj není vždy nejlepší řešení. Rozhodnutí zde záleží na účelu aplikace, kdy v případě potřeby maximálního výkonu či přístupu k funkcionalitě bude lepší využít sériový či paralelní vývoj oproti nástrojům pro multiplatformní vývoj aplikací.

2.1.1 Sériový vývoj aplikací

Sériový vývoj znamená, že se jednotlivé verze aplikace pro různé platformy budou vyvíjet za sebou. Tým vývojářů bude pracovat vždy na jedné platformě a po ukončení vývoje na danou platformu se vývoj přesune na platformu další. To s sebou přináší značnou prodlevu mezi distribucí aplikace pro další platformy a tým vývojářů musí být schopen pracovat se všemi použitými platformami. Zde nastává problém s opravami či aktualizacemi, kdy se musí vývojář vracet zpět na předchozí platformy pro vyřešení situace.

2.1.2 Paralelní vývoj aplikací

Paralelní vývoj znamená, že se verze aplikace pro různé platformy bude vyvíjet současně. Tým vývojářů je zde nutné rozdělit tak, aby část pracovala na aplikaci pro danou platformu. Výhodou je identický čas distribuce všech verzí aplikace pro všechny platformy najednou. Každá část vývojářů zde potřebuje znát pouze vlastní platformu, na které pracuje a další opravy v tomto případě může řešit kdokoliv, kdo se na práci s danou platformou podílel. Značnou nevýhodou je nutnost znásobení velikosti týmu, aby byla zajištěna stejná rychlost vývoje jako u platforem u sériového vývoje aplikací.

2.1.3 Vývoj za pomocí nástrojů

Vývoj za pomocí nástrojů znamená, že se vývoj mobilní aplikace provádí pouze jednou za použití specializovaného nástroje pro multiplatformní vývoj. Výstupem pak je samostatná aplikace pro každou platformu. Takový přístup snižuje náklady na vývoj oproti sériovému či paralelnímu vývoji, jelikož stačí pouze jeden tým, který bude umět pracovat s daným nástrojem a práci s cílovými platformami nemusí vůbec znát. Nevýhodou je zde menší výkon oproti nativním aplikacím a částečně omezený přístup k funkcím cílového zařízení.

3 Nástroje

Nástroje byly vybrány tak, aby obsáhly co největší počet programovacích jazyků.

- Android Studio [2]
  Android Studio nepatří mezi nástroje pro multiplatformní vývoj aplikací, ale bylo zde použito jako základ sloužící k porovnání jednotlivých nástrojů.
- Codename ONE [3]
  Java
- Gluon [4]
  Java, založený na knihovně JavaFX
- PhoneGap [5]
  JavaScript, založený na sadě nástrojů Apache Cordova
- Expo [6]
  TypeScript
- Ionic [7]
  TypeScript
- Felgo [8]
  QML
- Xamarin [9]
  C#
- Corona [10]
  Lua

3.1 Felgo

Felgo je soubor nástrojů pro vývoj multiplatformních aplikací založený na Qt frameworku. K práci s tímto nástrojem slouží vývojové prostředí Qt Creator. Jako hlavní programovací jazyk slouží QML, což je jazyk syntaxí velmi podobný klasickému JavaScriptu, ale podléhá jistým omezením. V případě, že je nutné psát platformě závislý kód, je možné využít programovacího jazyka C++.

3.1.1 Průběh

Felgo před vytvořením projektu vyžaduje registraci na oficiálních webových stránkách nástroje a následné přihlášení ve vývojovém prostředí. Všechny úkony potřebné pro vývoj mobilní aplikace je možné využít Qt Creatoru a není nutné využít žádné externí služby. Pro výsledné sestavení aplikace je však nutná konfigurace nástroje pro danou výstupní platformu. Projekt je rozdělen na zdrojové CPP a QML soubory. Tvorba logiky aplikace i uživatelského rozhraní se provádí skrze QML, a v případě výkonnostně náročného kódu také za pomocí zdrojových CPP souborů. Výstup z C++ kódu je pak možné provázat s QML. Felgo v základu obsahuje většinu standardních funkcí, které je možné na moderních mobilních zařízení očekávat. V případě nutnosti rozšíření funkcionality je možné využít pluginů.

3.1.2 Nástroje

Felgo poskytuje několik nástrojů, které nám vývoj aplikace usnadní.

3.1.2.1 Felgo Live Server je součástí instalace Felgo SDK. Jedná se o aplikaci, skrze kterou je možné testovat vyvíjenou mobilní aplikaci na počítači či na mobilním zařízení během samotného vývoje. Takto testovaná mobilní aplikace se pak automaticky obnoví po sestavení aplikace, tudíž je ihned možné vidět změny.

V defaultním nastavení se při sestavení vyvíjené aplikace spustí desktopový klient, který slouží jako simulátor. V případě, že chceme testovanou aplikaci vyzkoušet také na mobilním zařízení, je nutná instalace mobilní aplikace Felgo Live. Tento nástroj slouží také jako standardní výstup konzole pro debug či pro vzniklá chybová hlášení.

3.1.2.2 Felgo Live je mobilní aplikace, která nám dovolí mobilní zařízení propojit s Felgo Live Server a otestovat vyvíjenou aplikaci přímo na zařízení. Aplikace si také uchovává poslední testované aplikace, takže je možné aplikaci spustit i bez využití Felgo Live Server.

3.1.2.3 Designer je součást Qt Creatoru a je zde možné jednoduše vytvořit uživatelské rozhraní bez použití zdrojového kódu. Všechny úpravy se zde přenášejí oběma směry do a ze souborů QML. To znamená, že změna v souborech QML změní uživatelské rozhraní v designéru a naopak. Designér obsahuje velké množství ovládacích prvků.

3.1.3 Hodnocení

Felgo nabízí dostatečné množství dokumentace na webových stránkách nástroje. Kromě dokumentace pro samotný jazyk QML, je také dostupné množství dokumentace pro práci s nástrojem v jazyce C++. Součástí jsou také malé ukázky kódu. Kromě webových stránek je dokumentace dostupná také přímo ve vývojovém prostředí Qt Creator. Bohužel tento způsob dokumentace není možné využít vždy, neboť se často objevují chybové zprávy typu ‚dokumentace nebyla nalezena'.

Felgo SDK je rozhodně jedním z lepších nástrojů pro vývoj multiplatformních aplikací, ale bohužel jej nemohu zařadit mezi nástroje, které bych doporučil. K nevýhodám patří zejména krkolomná konfigurace sestavení a programovací jazyk QML. Ten je sice použitelný, ale chybí v něm mnoho funkcí z klasického JavaScriptu a jeho výkon také postrádá.

3.2 Codename One

Codaname One je nástroj pro vývoj multiplatformních aplikací založený na jazyce Java. Jedná se o rozšíření do několika populárních vývojových prostředí, jakými jsou Eclipse nebo NetBeans.

3.2.1 Průběh

Vývoj mobilní aplikace za pomocí tohoto nástroje je možné provést kompletně ve vývojovém prostředí. V případě, že chceme mobilní aplikace sestavit lokálně, je nutné manuálně stáhnout všechny potřebné soubory (SDK) a patřičně nakonfigurovat rozšíření. V opačném případě je možné využít cloudové řešení, které nástroj nabízí. Před využitím tohoto řešení je však nutné se zaregistrovat a přihlásit. Sestavit v cloudu je pak jednoduchou záležitostí, kdy je ve vývojovém nástroji tlačítko, které po stisknutí odešle kompletní projekt na server, který projekt sám sestaví. Výslednou sestavenou mobilní aplikaci je pak nutné stáhnout z webové stránky manuálně. K programování logiky zde slouží programovací jazyk Java, který je oproti klasickému jazyku Java omezen. Omezení se vztahuje na část nativních funkcí jazyka. Uživatelské rozhraní je možné vytvořit přímo v kódu nebo za pomoci designéru, které nástroj nabízí.

3.2.2 Nástroje

3.2.2.1 Designer je aplikace, která je součástí Codename One, ve které je možné nakonfigurovat uživatelské rozhraní. Rozhraní designéru není příliš propracované a zaměřuje se zejména na ovládací prvky. Na první pohled může působit velmi chaoticky, protože nabízí dva různé typy uživatelského rozhraní a neposkytuje téměř žádné informace.

3.2.2.2 Simulator je součást rozšíření, díky kterému je možné vyzkoušet vyvíjenou mobilní aplikaci. Simulátor nabízí sadu nástrojů, které lze využít pro diagnostiku výkonu či simulaci jednotlivých funkcí mobilních zařízení. Simulátor ale není možné použít jako kompletní náhradu mobilního zařízení či emulátoru.

3.2.2.3 Sestavení v cloudu je možnost automatického sestavení vyvíjené mobilní aplikace. Odeslání i sestavení projektu je možné provést za pomoci jednoho tlačítka. Aplikace se sestaví a následně je ji nutné stáhnout z webové stránky nástroje. Sestavení je možné provést pro několik cílových platforem.

3.2.3 Hodnocení

Dokumentace je u Codename One tvořena výhradně za pomocí utility Javadoc. K dispozici jsou ale také návody formou článků s úryvky kódu. Dokumentace jako taková je bohužel nepřehledná a neposkytuje mnoho informací.

Nástroj samotný poskytuje velkou míru abstrakce a díky tomu je většina funkcionality uschována za několika řádky kódu. Nevýhodou takovéto abstrakce je ale míra upravitelnosti. Další z nevýhod tohoto nástroje je omezení funkcionality jazyka Java, kdy je mnoho důležitých funkcí nedostupných (příkladem může být přesnější měření času, či postrádání jakékoliv manipulace se streamy). Problémem bylo také sestavení aplikace za pomoci cloudu, kdy nástroj ani stránka neposkytuje ani minimum informací, které bychom mohli od takové služby očekávat.

V případě, že chceme mobilní aplikace sestavit lokálně, je nutné manuální stažení potřebných souborů a následná konfigurace vývojového prostředí pro sestavení aplikace na danou platformu. Kromě náročnosti tohoto úkonu se tento způsob sestavení oficiálně nedoporučuje. Všechny přiložené nástroje jsou také velmi vzhledově jednoduché a špatně se v nich orientuje. Při shrnutí všech částečných hodnocení není možné tento nástroj doporučit s dobrým svědomím. Ačkoliv je práce v něm jednoduchá (po dostatečném bádání), nástroj zkrátka nedosahuje kvalit, které bychom mohli od moderních nástrojů pro multiplatformní vývoj mobilních aplikací očekávat.

3.3 Android Studio

Android Studio je oficiální vývojové prostředí pro vývoj mobilních aplikací pro platformu Android. Vývojové prostředí je založené na IntelliJ a používá jazyk Java nebo Kotlin společně s formátem XML. K vývoji je také možné použít jazyk C++.

3.3.1 Průběh

Všechny kroky vývoje mobilní aplikace je možné provést uvnitř vývojového prostředí. Pro sestavení či test mobilní aplikace v emulátoru není zapotřebí žádné manuální konfigurace či stahování potřebných souborů. Android Studio vše nabízí skrze jednoduché nástroje, které zajistí vše potřebné skrze pár kliknutí.

Kód aplikace a zdrojové soubory existují odděleně ve svých složkách. Každý zdrojový soubor je navíc označen vlastním indexem, díky kterému je možné soubor jednoduše použít uvnitř kódu. Tvorbu uživatelského rozhraní je také možné provádět přímo skrze XML soubory nebo díky přibalenému designéru. Aplikaci je možné okamžitě sestavit a spustit v emulátoru či na cílovém zařízení.

3.3.2 Nástroje

3.3.2.1 Designer je uživatelsky jednoduchý, ale komplexní designér uživatelského rozhraní. Tento designér je možné také přepnout do textového režimu, kdy máme přístup přímo k XML. Vlastní prvky uživatelského rozhraní je nutné vložit do uživatelského rozhraní přímo skrze XML a až pak se objeví v grafickém režimu designéru. Pak je s nimi možné volně pracovat jako s jakýmikoliv jinými prvky.

3.3.2.2 Emulator je integrovaný přímo do vývojového prostředí a je možné v něm spustit jakýkoliv obraz operačního systému Android. Kromě samotného obrazu také emulátor poskytuje řadu nastavení, kterými je možné emulované zařízení upravit podle své potřeby. Emulátor vyžaduje pro svou funkci povolení virtualizace s dalšími podpůrnými funkcemi. V případě, že není možné tyto funkce využít, je nutné použít x86 variantu obrazu operačního systému Android. V tomto případě dochází k podstatné ztrátě výkonu emulátoru a tento přístup se, pokud možno, nedoporučuje.

3.3.2.3 Virtual Device Manager slouží pro správu nainstalovaných zařízení pro emulátor. Umožňuje stažení libovolné verze operačního systému Android. Zde je také možné virtuální zařízení upravit, vyčistit či přímo spustit (nemusí jít pouze o testovací aplikace).

3.3.2.4 SDK Manager slouží pro správu nainstalovaných SDK. Umožňuje jejich stažení pro jakoukoliv verzi operačního systému Android a také stažení dalších nástrojů.

3.3.2.5 Device File Explorer slouží pro prozkoumání cílového mobilního zařízení. Toto zařízení musí být připojeno k počítači a musí být patřičně nastaveno.

3.3.2.6 Logcat slouží pro výpis protokolu událostí z připojeného mobilního zařízení nebo ze spuštěného emulátoru. Tento nástroj je také možné využít mimo vývojové prostředí v příkazové řádce pomocí programu adb.

3.3.2.7 Profiler nám umožňuje vytvořit částečně interaktivní záznam aktivity zařízení ve sledované době. Je možné zobrazit graf volání funkcí, různé grafy využití a další. Jeho použití s sebou ale přináší částečnou ztrátu výkonu testované aplikace.

3.3.3 Hodnocení

K Androidu existuje obrovské množství oficiální dokumentace společně s útržky i kompletními kusy kódu. Kromě online dokumentace existuje také nápověda, integrovaná do vývojového prostředí, ale její obsah se musí před zobrazením individuální stáhnout. Je možné najít návody pro všechny kategorie uživatelů.

Pokud pomineme fakt, že se jedná o oficiální prostředek k vývoji mobilních aplikací pro platformu Android, tak se jedná o nejlepší z testovaných nástrojů. Obsahuje vše potřebné s dostatečným množstvím dokumentace. Podpora nástroje je také velmi rozsáhlá. Jednoznačnou výhodou je jednoduché a automatické stažení všech potřebných souborů, což žádný z ostatních nástrojů nenabízí. Mezi nevýhody patří větší nároky vývojového prostředí a časté dočasné zamrzání počítače. Tyto problémy jsou znatelné, ale rozhodně se nejedná o stav, že by bylo Android Studio nepoužitelné.

3.4 PhoneGap

Adobe PhoneGap je nástroj pro vývoj hybridních mobilních aplikací od firmy Adobe. Základem nástroje je sada nástrojů Apache Cordova. Vývoj se provádí za pomoci standardních webových technologií, jakými jsou HTML5, CSS a JavaScript.

3.4.1 Průběh

Nástroj Adobe PhoneGap neposkytuje žádné vývojové prostředí a je možné využít k vývoji jakýkoliv textový editor. Pro počáteční vytvoření projektu je možné použít nástroj v CLI nebo aplikaci PhoneGap Desktop. Takto vytvořený nástroj je pak možné dále konfigurovat za pomocí nástroje v CLI. Desktopová aplikace pak dále slouží jako webový server, skrze který je možné aplikaci testovat (jak na mobilním zařízení, tak v prohlížeči). Obsah webového serveru se aktualizuje s uložením textového editoru. V případě, že chceme mobilní aplikaci testovat na mobilním zařízení, je nutné nainstalovat na cílové mobilní zařízení aplikaci PhoneGap Developer.

Sestavení aplikace je možné provést za pomocí CLI, ale v tomto případě je nutné manuálně stáhnout všechny potřebné soubory. Alternativou je využití sestavení v cloudu, které Adobe poskytuje pod názvem PhoneGap Build. Projekt zde stačí zabalit do archivu ZIP a nahrát na webové stránky nástroje. Aplikace se pak sestaví pro nakonfigurované platformy a sestavenou aplikaci je možné z webové stránky stáhnout.

Všechna funkcionalita je roztříděna do menších pluginů, které lze skrze CLI importovat do projektu.

3.4.2 Nástroje

3.4.2.1 PhoneGap Desktop je aplikace pro počítač. Slouží převážně jako správce projektů a jako webový server pro testování mobilní aplikace. Aplikaci stačí jednou spustit a další akce nejsou potřeba, protože se obsah serveru automaticky obnoví po provedení změn v kódu. Testovanou mobilní aplikaci je pak možné zobrazit v prohlížeči, desktopovém či mobilním, nebo skrze mobilní aplikaci PhoneGap Developer.

3.4.2.2 PhoneGap Developer je mobilní aplikace sloužící pro připojení k aplikaci PhoneGap Desktop.

3.4.2.3 PhoneGap CLI je základní nástroj v příkazové řádce sloužící k manipulaci s projektem. Nabízí mnohem širší obsah funkcionality než aplikace PhoneGap Desktop. Manipulace s projektem, instalace rozšíření či lokální sestavení mobilní aplikace se provádí skrze tento nástroj.

3.4.2.4 PhoneGap Build je cloudový nástroj pro automatické sestavení mobilní aplikace. Je zde nutné se zaregistrovat a přihlásit před samotným použitím nástroje. Pro sestavení mobilní aplikace stačí nahrát kód aplikace v archivu ZIP na webovou stránku nástroje a aplikace se sestaví automaticky, bez nutnosti další konfigurace. Po úspěšném sestavení je pak možné sestavenou mobilní aplikaci stáhnout pro nakonfigurované platformy (pro které je mobilní aplikace cílena). K dispozici je také výstup protokolu událostí sestavení.

V případě, že přijdeme o zdrojový kód aplikace je také možné stáhnout zpět archiv ZIP. Tento kód se ale při každé nové verzi nahrané mobilní aplikace přepíše.

3.4.3 Hodnocení

Z hlediska dokumentace PhoneGap nabízí krátký návod pro začátečníky i se zdrojovým kódem. Podrobnější dokumentace je pak dostupná přímo na webových stránkách Apache Cordova. Tato dokumentace je velmi podrobná a obsahuje množství ukázek daných funkcí.

PhoneGap je dostatečně propracovaný nástroj a je možné ho doporučit bez problémů. Jediným problémem či výtkou může v tomto případě být nedostatečná funkcionalita nástroje PhoneGap Destkop, ale tu je možné nahradit několika málo řádky v CLI. Sestavení mobilní aplikace za pomocí cloud služby bylo pokaždé rychlé a proběhlo bez problémů. Testování mobilní aplikace také fungovalo bez problémů v prohlížeči i za pomoci aplikace PhoneGap Developer.

3.5 Xamarin

Xamarin je nástroj pro vývoj generovaných multiplatformních mobilních aplikací od firmy Microsoft. Jedná se o volitelnou součást vývojového prostředí Microsoft Visual Studio. Vývoj se zde provádí za pomoci jazyka C# a uživatelské rozhraní je možné tvořit skrze XAML (forma XML).

3.5.1 Průběh

Projekt mobilní aplikace je rozdělen na několik individuálních projektů, které se nacházejí ve společném řešení. Jako jádro aplikace slouží jeden projekt a další projekty slouží pro kód, který je závislý na dané platformě. Při sestavení se pak sloučí projekt základní s projektem pro danou platformu. Ostatní projekty je nutné provázat pomocí Dependency Service, kdy nadefinujeme v základním projektu funkcionalitu a Xamarin pak na základě dané platformy vybere patřičnou implementaci dané funkcionality.

Kromě kódu má také každý platformě závislý projekt vlastní sadu zdrojových souborů, takže je nutné mít soubory několikrát na více místech. Vývoj aplikace se kompletně provádí přímo ve vývojovém prostředí a není nutné využít žádné externí prostředky. Xamarin také poskytuje plně funkční emulátor a potřebné nástroje (podobné těm u Android Studia). Kromě toho je také možné mobilní aplikaci přímo nainstalovat na připojené mobilní zařízení.

Vývojové prostředí Visual Studio poskytuje možnost nainstalovat rozšíření pro vývojové prostředí. Kromě toho je také možné nainstalovat širokou řadu rozšíření pro samotný nástroj Xamarin, které je pak možné volně využít při tvorbě mobilní aplikace. Tato rozšíření nemusí pocházet přímo z oficiálních zdrojů, ale je možné je získat i od jiných vývojářů.

3.5.2 Nástroje

3.5.2.1 Designer je součástí Xamarinu a je rozdělen podle cílových platforem. Uživatelské rozhraní je možné upravit v grafickém režimu nebo přímo v souboru XAML.

3.5.2.2 Emulator je integrován přímo do vývojového prostředí a umožňuje spustit jakýkoliv nainstalovaný obraz operačního systému Android. Stejně jako u Android Studia, je dobře konfigurovatelný podle našich požadavků. K funkci je potřeba virtualizace a další nástroje, kdy v případě chybějících nástrojů je nutné využít x86 varianty obrazu operačního systému Android. V tomto případě dochází k podstatné ztrátě výkonu emulátoru a tento postup se, pokud možno, nedoporučuje.

3.5.2.3 Simulator je alternativa pro emulátor, pokud chceme otestovat aplikaci běžící pod iOS.

3.5.2.4 Android Device Manager slouží pro správu nakonfigurovaných zařízení. Je možné jednotlivé virtuální zařízení upravit, smazat či vytvořit. Kromě toho je možné dané zařízení také pouze spustit (bez testu mobilní aplikace).

3.5.2.5 Android SDK Manager slouží pro správu souborů potřebných pro sestavení mobilní aplikace. Je možné zde jednoduše stáhnout všechny potřebné soubory nebo je aktualizovat.

3.5.2.6 ADB CLI slouží pro práci se zařízením skrze příkazovou řádku. Obsahuje velké množství funkcí, které lze pro práci se zařízením s operačním systémem Android použít. Mezi ně se může řadit zobrazení protokolu událostí, simulace doteku či další manipulace s mobilním zařízením.

3.5.2.7 Android Device Monitor slouží pro analýzu mobilního zařízení během testu vyvíjené mobilní aplikace. Je podobný nástroji Profiler z vývojového prostředí Android Studio.

3.5.2.8 Device Log (Logcat) slouží k zobrazení protokolu událostí připojeného zařízení, ať se jedná o skutečné mobilní zařízení nebo emulátor. Je dostupný pro platformy Android i iOS.

3.5.3 Hodnocení

Xamarin nabízí rozsáhlé množství oficiální dokumentace. Součástí jsou také návody obsahující úryvky i kompletní segmenty kódu. Díky rozdělení projektů je možné získat mnohem širší přístup k nativním funkcím mobilního zařízení. Nevýhodou je nutnost použití Dependency Service, kde je jednoduché se ztratit.

Xamarin je ze všech úhlů pohledu velmi kvalitní nástroj, který je možné určitě doporučit.

3.6 Expo

Expo je nástroj pro vývoj multiplatformních aplikací využívající nástroj React Native. Pro vývoj je použita obdoba programovacího jazyka JavaScript s názvem TypeScript. Tento jazyk má sadu typů a nedovoluje takovou volnost jako klasický JavaScript.

3.6.1 Průběh

Expo neposkytuje žádné vývojové prostředí a je tedy možné použít libovolný textový editor. Základním kamenem tohoto nástroje je jeho CLI, kdy se všechny akce s projektem provádějí právě skrze CLI. Projekty jsou rozděleny na dva základní typy (čistý React Native nebo managed). Čistý React Native projekt využívá své vlastní knihovny a je nutné ho dále konfigurovat. Druhá managed možnost využívá také své vlastní knihovny. Obě verze nejsou navzájem kompatibilní a mohou se zásadně lišit.

Testování je možné provést pomocí emulátoru pro Android, simulátoru pro iOS nebo s využitím mobilního nástroje Expo pro testování vyvíjené aplikace přímo na mobilním zařízení. Sestavení vyvíjené mobilní aplikace je možné provést lokálně, ale je nutné ručně sestavení nakonfigurovat a stáhnout všechny potřebné soubory. Druhou variantou je využití cloud řešení sestavení, které nástroj Expo poskytuje.

CLI spolupracuje spolu s nástrojem NPM a díky tomu je možné rozšíření jednoduše stáhnout i importovat do projektu. Knihovny pro oba typy projektů existují odděleně a není tedy možné použít knihovnu pro managed projekt v čistém React Native projektu a naopak.

3.6.2 Nástroje

3.6.2.1 Expo CLI slouží pro většinu práce s projektem. Je možné zde vytvořit projekt, konfigurovat ho, stáhnout knihovny a importovat je do projektu. Skrze tento nástroj se také spouští všechny další nástroje, které nástroj Expo poskytuje.

3.6.2.2 Expo Developer Tools běží v prohlížeči a funguje jako jednoduché rozhraní pro test mobilní aplikace. Umožňuje testovanou mobilní aplikaci spustit v emulátoru, simulátoru či odeslat na připojené mobilní zařízení (toto platí pouze pro mobilní zařízení s operačním systémem Android). Takto spuštěná aplikace se pak vždy obnoví při změně kódu.

Nástroj také nabízí protokol událostí, ale jeho obsah je omezen na standardní konzolový výstup a chybová hlášení. Žádné podrobnější informace nenabízí.

3.6.2.3 Expo Developer Services je sada cloud nástrojů pro nástroj Expo. Nástroj umožňuje sestavit aplikaci.

3.6.3 Hodnocení

Expo poskytuje dostatečnou míru dokumentace k nástroji samotnému i k jeho rozšířením. U každého rozšíření pak existuje jednoduchý návod na instalaci daného rozšíření a úryvky kódu potřebné k jeho použití.

Expo je dobrý nástroj pro tvorbu multiplatformních aplikací i pro začátečníky. Všechny nástroje fungují bez problémů. V případě, že nechceme pokračovat ve využívání nástroje Expo, nástroj umožňuje vypuštění projektu na čistý React Native projekt. Mezi problémy se řadí nedostatečné hlášení a popis vzniklých chyb a nutnost stáhnout rozšíření pro každou sebemenší funkci. K příkladům například patří práce se soubory, baterie či senzory – kdy každá akce vyžaduje své vlastní rozšíření a ty nejsou součástí nástroje od začátku.

Nekompatibilita mezi rozšířeními pro managed projekt a čistý React Native projekt také není dobře oznámena.

3.7 Corona

Corona je nástroj pro vývoj multiplatformních aplikací od CoronaLabs. Nástroj je zaměřen převážně na tvorbu her, ale je možné ho využít i pro běžné mobilní aplikace. Vývoj v tomto nástroji se provádí za pomoci jazyka Lua, ale je možné ho rozšířit o kód v jazycích Java nebo Objective-C.

3.7.1 Průběh

Nástroj neposkytuje žádné vývojové prostředí, takže je vývoj možné provádět v libovolném textovém editoru. Pro jisté textové editory ale existuje rozšíření právě pro tento nástroj.

K práci s projektem slouží Corona Simulator, kdy vytvoření projektu a následné spuštění aplikace se provádí právě zde. Sestavení mobilní aplikace je možné provést lokálně, ale je nutné provést manuální konfiguraci a stáhnout všechny potřebné soubory. K dispozici je živé sestavení, kdy je možné využít Corona Live Server k následnému odeslání změn v aplikaci na cílové zařízení.

3.7.2 Nástroje

3.7.2.1 Corona Simulator slouží jako centrální prvek toto nástroje. Jeho cílem je vytvoření, správa a testování samotné mobilní aplikace.

3.7.2.2 Corona Live Server je aplikace sloužící pro distribuci změn v kódu mobilní aplikace mezi všechna živá sestavení vyvíjené mobilní aplikace. Takto sestavenou mobilní aplikaci je pak možné nainstalovat na libovolný počet mobilních zařízení a aplikace bude následně udržována vždy aktuální k aktuálnímu projektu.

3.7.3 Hodnocení

Corona nabízí velmi obsáhlou dokumentaci společně s návody na použití tohoto nástroje. Tyto návody mají za cíl vytvoření kompletní aplikace pomocí podrobného tutoriálu s úryvky kódu. Zdrojový kód je pak možné si rovnou stáhnout.

Nevýhodou tohoto nástroje je využití programovacího jazyka Lua, což z určitého pohledu limituje možnosti vývojáře. Tento jazyk není tak kvalitní, v obsahu funkcionality nebo syntaxe, jako ostatní jazyky. Z tohoto důvodu není možné tento nástroj doporučit, pokud programovací jazyk přímo nevyhledáváme.

3.8 Ionic

Ionic je nástroj pro vývoj multiplatformních aplikací za pomocí jazyka TypeScript, což je odnož jazyka JavaScript. Tento nástroj je založen na nástrojích React a Angular.

3.8.1 Průběh

Vývoj v tomto nástroji je možné provádět pomocí přiloženého Ionic Studia nebo pouze za použití CLI nástroje.

Sestavení mobilní aplikace u tohoto nástroje probíhá netradičně a spoléhá se na nástroje daných platforem. Při sestavení se jednoduše otevře projekt v Android Studiu (pokud se jedná o sestavení pro platformu Android), nebo v programu Xcode (pokud se jedná o sestavení pro platformu iOS). Sestavení pro platformu iOS lze tedy provádět pouze na zařízení Apple.

3.8.2 Nástroje

3.8.2.1 Ionic CLI je základní nástroj pro manipulaci s projektem a jeho sestavení. Je možné jej využít, pokud nechceme využít Ionic Studio, neboť plní identickou funkci.

3.8.2.2 Ionic Studio je jednoduché vývojové prostředí. Jedná se o volitelnou součást nástroje Ionic.

3.8.2.3 Ionic Appflow je cloud služba, která umožňuje sestavení mobilní aplikace v cloudu a odeslání nejnovějších verzí aplikace na mobilní zařízení. Služba vyžaduje registraci a přihlášení.

3.8.3 Hodnocení

K nástroji Ionic existuje velké množství dokumentace pro všechny varianty nástroje (React, Angular či Vue). Dokumentace je vedena formou návodů, kdy se návod zaměřuje na určité použití a popisuje společně s útržky kódu všechny potřebné kroky.

Stejně jako u nástroje Expo, tento nástroj také využívá TypeScript. Z toho důvodu je mírně omezen. Celkově je tento nástroj možné doporučit.

3.9 Gluon

Gluon je nástroj pro vývoj multiplatformních aplikací od tvůrců JavaFX. Nástroj existuje jako rozšíření do populárního vývojového prostředí Eclipse a vývoj probíhá za pomoci programovacího jazyka Java a FXML.

3.9.1 Průběh

Práce s nástrojem Gluon je identická jako s obyčejným JavaFX. V tomto případě je ale k dispozici několik specifických prvků uživatelského rozhraní. Uživatelské rozhraní je možné vytvořit buď skrze kód, nebo v designéru. Ten není součástí nástroje a je nutné ho stáhnout a nainstalovat zvlášť. Uživatelské rozhraní vytvořené pomocí designéru je obsaženo v souborech FXML, tyto soubory je pak možné v kódu vložit jako jakýkoliv jiný prvek uživatelského rozhraní.

Sestavení a další operace je možné provést přímo ve vývojovém prostředí pomocí připravených gradle skriptů. Je takto možné mobilní aplikaci sestavit pro dané platformy či ji nainstalovat na připojené mobilní zařízení. Mobilní aplikace také může obsahovat moduly, které poskytují přístup k nativním funkcím mobilních zařízení. Tyto funkce lze jednoduše přidat či odebrat pomocí konfiguračního okna.

3.9.2 Nástroje

3.9.2.1 Scene Builder slouží pro vytvoření uživatelského rozhraní pro mobilní aplikaci. Je to varianta klasického Scene Builderu pro JavaFX, kdy tento nástroj obsahuje prvky uživatelského rozhraní, které se vyskytují pouze u mobilních zařízení.

3.9.3 Hodnocení

K tomuto nástroji je možné najít průměrné množství dokumentace. Bohužel, díky malému rozšíření tohoto nástroje, neexistuje dost návodů, které by bylo možné využít. Některé informace se nacházejí pouze v příspěvcích fór v jednočíselných počtech.

Ačkoliv se tedy jedná o solidní nástroj, není možné ho doporučit.

4 Netestované nástroje

4.1 JUniversal

JUniversal[11] je malý nástroj pro překlad programů v programovacím jazyce Java do jiných jazyků (C# nebo Objective-C). K nástroji také patří knihovna JSimple, která se stará o přístup k nativním funkcím mobilních zařízení. Nástroj se nijak nepodílí na vývoji mobilní aplikace, ale slouží pouze jako mezikrok sestavení, kdy přeloží kód od jiného programovacího jazyka. Toto je možné díky podobnosti jazyků a jeden řádek v originálním kódu pak odpovídá jednomu řádku přeloženého kódu.

K nástroji existuje dokumentace tvoření za pomoci nástroje Javadoc.

4.2 Sencha JS

Sencha JS[12] je nástroj pro vývoj multiplatformních aplikací. Je založený na klasických webových technologiích HTML5, CSS a JavaScript. Nástroj slouží pro rychlé a jednoduché vytvoření mobilní aplikace za použití mnoha předdefinovaných prvků uživatelského rozhraní. Tento nástroj je dostupný pouze v placené variantě.

Kromě samotného nástroje je k dispozici také Sencha Architect, který slouží k vytvoření uživatelského rozhraní pouze za pomocí drag-drop techniky. Dále pak existuje Sencha Stencils pro rychlé prototypování uživatelského rozhraní.

4.3 Appcelerator

Appcelerator[13] je nástroj pro vývoj multiplatformních aplikací. Jedná se o nástroj podobný jako Sencha JS. Kromě samotného vývojového nástroje Appcelerator také poskytuje služby, které se zaměřují na analytiku využívání aplikace a dalších informací o koncovém uživateli.

5 Vybraný nástroj

Pro vývoj multiplatformních aplikací pro mobilní zařízení bohužel neexistuje dostatečné množství nástrojů, aby bylo možné z nich vybírat. Možnosti jsou v tomto případě pouze dvě, a to: Gluon a Codename One.

Codename One byl nástroj funkční, ale na první pohled velmi nedodělaný. Navigace a ovládání byly místy složité, a v případě problémů nebyly k dispozici žádné užitečné informace. Sestavení v cloudu je sice funkční, ale jedná se o vše, co služba nabízí. Kromě stažení sestavené aplikace se zde nenachází nic. Z toho důvodu byl Codename One vyřazen.

Druhým nástrojem na výběr byl Gluon, ten byl také nakonec vybrán pro zhotovení demonstrační aplikace. Tento nástroj sice není velmi rozšířený, ale celkově působí jako dopracovaný nástroj. Má dostatečné množství funkcí a snadno se ovládá.

5.1 Alternativa

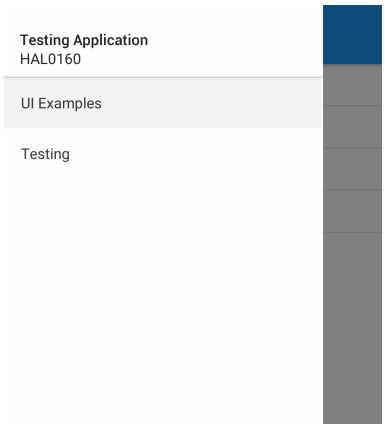
Je nutné dodat, že nástroj Gluon byl vybrán pouze z důvodu omezení použitého programovacího jazyka Java. V případě, že by toto omezení stanoveno nebylo, byl by vybrán Adobe PhoneGap. Ten byl ve všech ohledech bezproblémový, a v případě jakéhokoliv problému bylo k dispozici dost informací k jeho vyřešení. I když nenabízí designér, tak bylo vytvoření uživatelského rozhraní pomocí HTML a CSS velmi jednoduché.

5.2 Výkon a jazyk

Po výkonnostní stránce byl Gluon na stejném místě jako zbytek nástrojů využívající jazyky Java (zde je obsaženo i prostředí Android Studio). Použitá verze jazyka Java nebyla ničím omezena a bylo možné použít nejnovější programovací techniky.

6 Návrh demonstrační aplikace

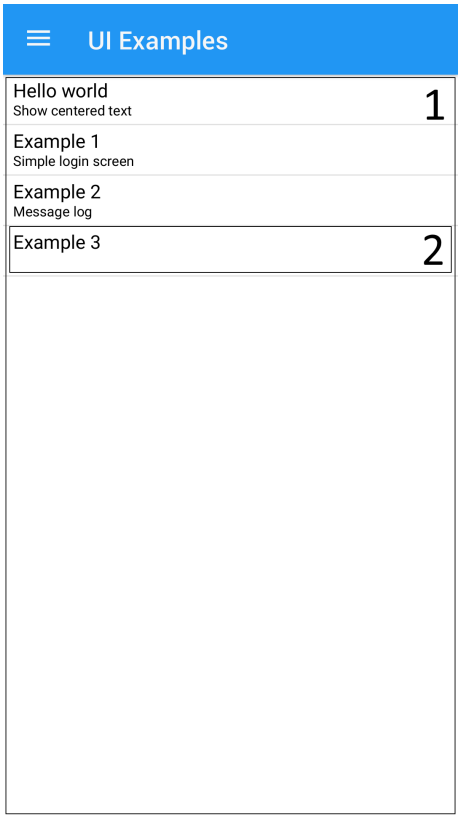
Hlavním účelem demonstrační aplikace je ukázka několika předdefinovaných ukázek uživatelského rozhraní. Kromě ukázek také aplikace obsahuje seznam testů, kde je možné testy vybrat a vybrané testy spustit. Výsledek testů se pak postupně zobrazí ve vlastním okně společně s průběhem. Na konci všech testů se zobrazí statistiky testů s možností zobrazit graf výsledků pro jednotlivé testy.



Obrázek 1: Úvodní Okno / Menu

6.1 Úvodní okno

Úvodní okno je rozděleno na dvě podokna, mezi kterými lze přepínat pomocí vysouvacího bočního menu. První podokno obsahuje seznam ukázek různých konfigurací uživatelského rozhraní. Druhé podokno obsahuje seznam testů, které lze pomocí aplikace spustit.



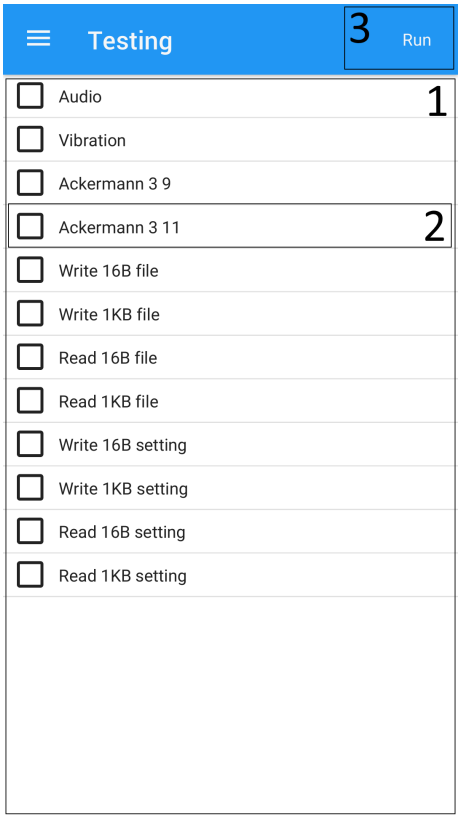
Obrázek 2: Seznam ukázek

6.1.1 Seznam ukázek

Seznam ukázek obsahuje seznam předdefinovaných ukázek uživatelského rozhraní. Klepnutím na jakoukoliv ukázku je možné danou ukázku zobrazit v novém okně.

1. Seznam ukázek
   V této oblasti se zobrazí všechny dostupné ukázky registrované v aplikaci.

2. Ukázka
   Element ukázky uživatelského rozhraní obsahuje název a popis obsahu ukázky. Klepnutím kdekoliv do těla ukázky se daná ukázka zobrazí v novém okně.



Obrázek 3: Seznam testů

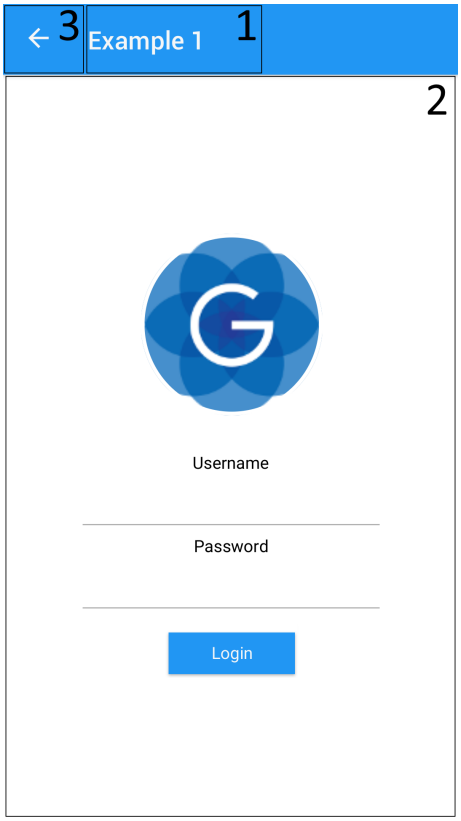
6.1.2 Seznam testů

Seznam testů obsahuje seznam testů, které jsou v aplikaci dostupné. Jednotlivé testy lze vybrat a následně spustit.

1. Seznam testů
   V této oblasti se zobrazí všechny testy, které aplikace nabízí.

2. Test
   Element testu obsahuje zaškrtávací tlačítko a popis testu. Tlačítko je možné zaškrtnout pomocí klepnutí do těla testu.

3. Tlačítko pro spuštění testů
   Stisknutím tohoto tlačítka se spustí série všech vybraných testů. V případě, že žádný test není vybrán, toto tlačítko nebude možné stisknout. Stisk tohoto tlačítka způsobí přepnutí do jiného okna.

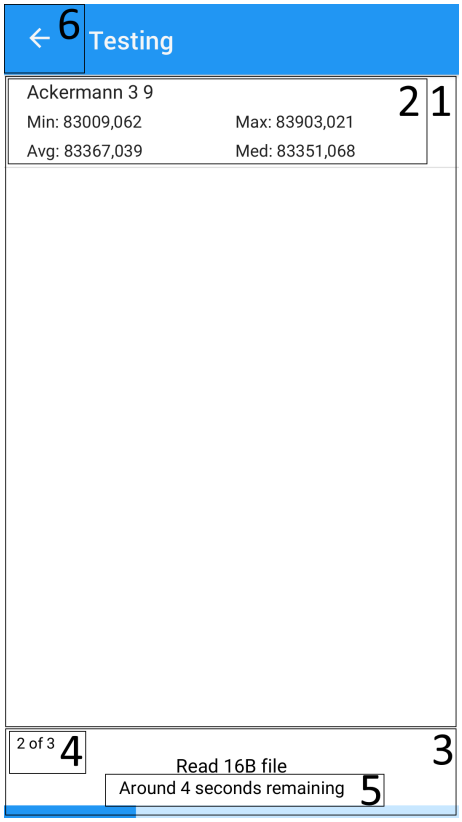


Obrázek 4: Ukázka

6.2 Ukázka

Toto okno obsahuje konkrétní ukázku uživatelského rozhraní. Ukázku je možné opustit pomocí tlačítka zpět. Žádný z použitých elementů uživatelského rozhraní nemá implementaci a nereaguje na vstup.

1. Název ukázky
2. Obsah ukázky
3. Tlačítko pro návrat
   Tímto tlačítkem je možné vrátit se zpět do hlavního okna.



Obrázek 5: Průběh sady testů

6.3 Průběh testů

Okno pro průběh testů obsahuje oblast pro seznam hotových testů a oblast obsahující momentálně běžící test. V případě, že je sekvence testů dokončena, tak se oblast pro právě běžící test skryje. Toto okno není možné opustit, dokud nebudou všechny testy dokončeny.

1. Seznam hotových testů
   Tento seznam obsahuje všechny testy, které byly ukončeny. V případě, že byly všechny testy ukončeny, je možné na daný test klepnout k zobrazení grafu s výsledky daného testu.

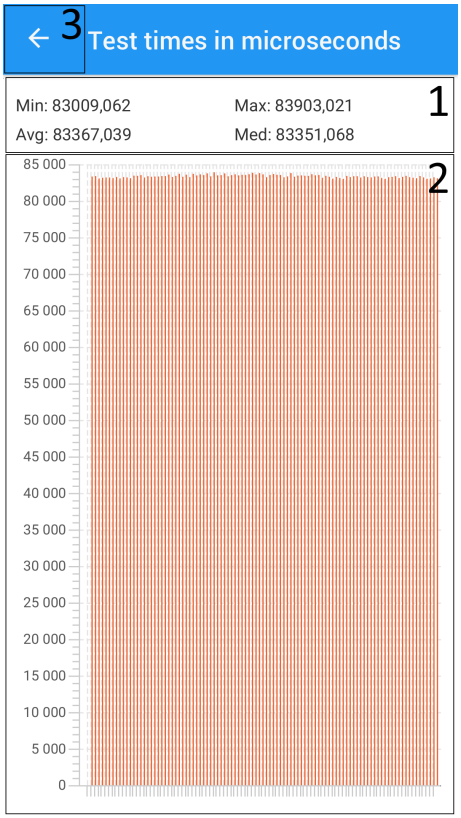
2. Zkrácený výsledek testu
   Tato oblast obsahuje zkrácený výsledek ukončeného testu. Zkrácený výsledek obsahuje minimum, maximum, průměr a medián výsledků testu.

3. Oblast právě běžícího testu
   Tato oblast je zobrazena pouze po dobu, po kterou testy běží. Obsahuje stav a úroveň dokončení běžícího testu.

4. Postup
   Toto číslo ukazuje pozici právě běžícího testu v sekvenci spuštěných testů.

5. Stav testu
   Tato oblast ukazuje pravděpodobný čas zbývající do dokončení.

6. Tlačítko pro návrat
   Tímto tlačítkem lze opustit obrazovku po dokončení všech testů.



Obrázek 6: Detail výsledku testu

6.4 Detail výsledku testu

Toto okno obsahuje podrobné informace o dokončeném testu. Lze jej kdykoliv opustit a navrátit se do okna s průběhem testů. Okno poskytuje jednoduchý sloupcový graf, který ukazuje jednotlivé výsledné časy instancí daného testu.

1. Výsledek testu
   Tato oblast obsahuje zkrácený výsledek testu. Mezi zobrazené hodnoty patří minimum, maximum, průměr, medián všech naměřených hodnot, dále také celkový čas testu.

2. Graf testu
   Tento sloupcový graf ukazuje jednotlivé výsledky instancí testu. K dispozici je jednoduché měřítko s minimem a maximem všech hodnot.

3. Tlačítko pro návrat
   Tímto tlačítkem lze opustit obrazovku po dokončení všech testů.

6.5 Testy v aplikaci

Demonstrační aplikace obsahuje několik testů. Tyto testy jsou víceméně shodné s testy, které byly prováděny u všech nástrojů.

6.5.1 Zápis do souboru

Tento test změří dobu, za kterou je možné zapsat řetězec do souboru. Po ukončení testu se soubor smaže. Tento test existuje ve dvou variantách – 1 KB a 1024 KB.

6.5.2 Čtení ze souboru

Tento test změří dobu, za kterou je možné přečíst řetězec ze souboru. Před zahájením testu se vytvoří textový soubor o dané velikosti. Tento test existuje ve dvou variantách – 1 KB a 1024 KB.

6.5.3 Zápis do preferencí

Tento test změří dobu, za kterou je možné uložit řetězec do preferencí. Po ukončení testu se uložený záznam smaže. Tento test existuje ve dvou variantách – 1 KB a 1024 KB

6.5.4 Čtení z preferencí

Tento test změří dobu, za kterou je možné přečíst řetězec z preferencí. Před zahájením testu se zapíše nový záznam o dané velikosti. Tento test existuje ve dvou variantách – 1 KB a 1024 KB.

6.5.5 Vibrace

Tento test změří dobu, za kterou je možné zahájit vibraci zařízení. Doba vibrací se do změřené doby nepočítá, jelikož jsou vibrace prováděny asynchronně.

6.5.6 Audio

Tento test změří dobu, za kterou je možné zahájit přehrávání zvukového souboru. Doba přehrávání se do změřené doby nepočítá, jelikož je zvuk přehráván asynchronně. Obsahem testu je i načtení zvukového souboru.

6.5.7 Ackermannova funkce

Tento test změří dobu, za kterou se spočítá Ackermannova funkce. Tento test existuje ve dvou variantách, a to s parametry 3-9 a 3-11.

6.6 Ukázky v aplikaci

Demonstrační aplikace obsahuje několik ukázek uživatelského rozhraní. Tyto ukázky jsou vytvořeny za pomocí FXML a neobsahují žádnou funkcionalitu. Jejich funkce je pouze pro přiblížení možností, které nástroj Gluon nabízí z hlediska designu uživatelského rozhraní.

6.6.1 Hello World

Tato ukázka obsahuje pouze ukázkový text umístěný uprostřed obrazovky.

6.6.2 Přihlašovací obrazovka

Tato ukázka obsahuje základní přihlašovací obrazovku s obrázkem a dvěma vstupními poli.

6.6.3 Seznam zpráv

Tato ukázka obsahuje jednoduchý seznám zpráv s ikonami.

6.6.4 Platba přes kartu

Tato ukázka obsahuje jednoduché uživatelské rozhraní, které bychom mohli najít při platbě kartou u většiny služeb.

7 Testování výkonu

7.1 Testované zařízení

Testy byly prováděny na zařízení se systémem Android, konkrétně na Xiaomi Mi A2 64 GB, model M1804D2SG. Jednalo se o Android verze 9 s aktualizací ze dne 5. listopadu 2019 (nejnovější dostupná aktualizace).

7.2 Testované nástroje

Pro testování výkonu byly použity nástroje PhoneGap, Xamarin, Felgo, Corona, Ionic, Gluon, Codename ONE, Expo a Android Studio, které sloužilo pro porovnání vůči nativní aplikaci. Všechny nástroje byly aktualizované a aplikace byly sestaveny pro Android 9.

7.3 Proces měření

Aby se zamezilo dalším externím vlivům, tak byly k měření času použity pouze funkce, které byly v daných nástrojích dostupné. V případě, že bylo možné použít více funkcí, byly použity ty s největší přesností a výsledné hodnoty byly následně zaokrouhleny na mikrosekundy.

Celý proces probíhal postupně, kdy se každé měření provedlo 100krát za sebou a stejně tak se měření pro všechny aplikace spouštěly postupně.

7.4 Testy

7.4.1 Soubory

Obsahem tohoto testu bylo čtení a zápis souborů. Testy byly provedeny dvakrát ve velikostech 1 KB a 1024 KB. V obou případech se jednalo o zápis a čtení řetězce. Obsahem testu byl kompletní proces zahájený vytvořením první proměnné a ukončený uzavřením souboru. U tohoto testu bylo nutné vyčkat na ukončení zápisu.

7.4.2 Preference

Obsahem tohoto testu bylo čtení a zápis řetězce do databázové struktury. Ze všech nástrojů byly vybrány metody, které akceptovaly klíč a hodnotu. V případě webových aplikací bylo využito lokální uložiště localStorage a v případě nativních aplikací se jednalo o Shared Preferences. Testy byly provedeny dvakrát ve velikostech 1 KB a 1024 KB. U tohoto testu bylo nutné vyčkat na ukončení zápisu.

U některých nástrojů se implementace preferencí liší a některé nástroje nebylo možné použít pro zápis 1024 KB velkých řetězců.

7.4.3 Vibrace

Obsahem tohoto testu bylo zahájení vibrací. Ve všech případech bylo využito defaultní nastavení, které daný nástroj poskytuje. U této funkce nebylo nutné vyčkat na ukončení vibrací, protože jsou vibrace řešeny asynchronně.

7.4.4 Zvuk

Obsahem tohoto testu bylo zahájení přehrávání zvuku společně s načtením zvukového souboru. Soubor byl poskytnut ve formátu mp3 a nebyl načtený předem. Audio objekt byl po každém přehrání zahozen a vytvořen nový, přičemž jakékoliv dočasné ukládání nebylo nijak řešeno.

7.4.5 Ackermann

Obsahem tohoto testu bylo spuštění Ackermannovy funkce. Tento test měl za cíl určit výkon nástroje i využitého programovacího jazyka. Tento test byl proveden celkově dvakrát, jednou s parametry 3-9 a podruhé s parametry 3-11.

7.5 Poznámka

V případech, kdy došlo k načtení zdrojů během spuštění mobilní aplikace, nebyl na tento fakt brán ohled. Testy byly prováděny slepě vzhledem k variacím v jednotlivých funkcích, které nástroj nabízí. V případě, že nebylo možné test provést za sebou, tak byl prováděn manuálně v krocích nebo nahrazen podobným testem (nebo testem jednodušším vzhledem k omezením nástroje).

8 Závěr

Všechny testované nástroje pro multiplatformní vývoj aplikací měly své výhody i nevýhody. Nelze tedy přímo říci, který z nich je nejlepší. Po nabytí zkušeností s každým z nich je ale možné určit, který je z nich pro naše účely vhodnější. Toto rozhodnutí bylo pouze částečně ovlivněno výsledky testů.

V kapitole 5 byla již většina věcí vysvětlena. Důvodem proč byl zvolen nástroj Gluon bylo pouze omezení na nástroje, které pracují v jazykem Java. V případě, že by toto omezení bylo uvolněno, tak by byl zvolen nástroj PhoneGap. Ten sice k nástrojům na prvních místech testů nepatří, ale kompenzuje tento nedostatek jednoduchou použitelností nástroje. V případě, že chybějící výkon potřebujeme, tak je lepší využít nativní aplikace nebo případně nástroje Xamarin nebo Gluon.

Výsledky jednotlivých testů a grafy je možné prohlédnout níže.

8.1 Výkon

Z výkonnostní stránky jsou na tom všechny nástroje dobře. Toto ovšem neplatí pro všechny nástroje, kdy například nástroj Expo potřeboval k práci s preferencemi mnohem více času než je tomu u ostatních nástrojů (Tabulky 7 a 9). Důvodem k těmto zpomalením je šifrování obsahu preferencí.

U zápisu a čtení souborů všechny nástroje odvedly dobrou práci a maximální doba operace se udržela pod 40 ms (Tabulky 11 a 13).

Navzdory tomu, že se nejedná o nativní aplikace, se nástrojům Ionic, Expo a Corona (Tabulka 15) podařilo spustit vibrace rychleji než ostatním nástrojům. Předpokládá se ale, že se jedná o non-blocking funkci, kdy se program nezastaví, ale pouze odešle požadavek na vibrace níže položené službě. Díky tomu je možné předpokládat, že doba operace bude podobná zbytku nástrojů.

Přehrání audia (Tabulka 16) mělo podobné výsledky jako vibrace, kdy u některých nástrojů došlo k nahrání zvukové stopy již během spuštění aplikace nebo došlo k pouze volání non-blocking funkce. Jedinou výjimkou je nástroj Expo, kdy k přehrání zvukové stopy aplikace potřebovala abnormálně dlouhou dobu.

Ackermannova funkce (Tabulka 17) slouží pro ukázku výkonu samotného jazyka. Je možné vidět, že nástroje Codename ONE, Expo a Felgo nedokončily funkci s parametry 3 / 11. Toto bylo způsobeno přetečením paměti stacku.

Literatura

1. *Fragmentace u mobilních aplikací*. Dostupné také z: https://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm.
2. *Android Developers* [online] [cit. 2020-03-22]. Dostupné z: https://developer.android.com/.
3. *Codename ONE* [online] [cit. 2020-03-22]. Dostupné z: https://www.codenameone.com/.
4. *Gluon* [online] [cit. 2020-03-22]. Dostupné z: https://gluonhq.com/.
5. *PhoneGap* [online] [cit. 2020-03-22]. Dostupné z: https://phonegap.com/.
6. *Expo* [online] [cit. 2020-03-22]. Dostupné z: https://expo.io/.
7. *Ionic* [online] [cit. 2020-03-22]. Dostupné z: https://ionicframework.com/.
8. *Felgo* [online] [cit. 2020-03-22]. Dostupné z: https://felgo.com/.
9. *Xamarin* [online] [cit. 2020-03-22]. Dostupné z: https://dotnet.microsoft.com/apps/xamarin.
10. *Corona Labs* [online] [cit. 2020-03-22]. Dostupné z: https://coronalabs.com/.
11. *JUniversal* [online] [cit. 2020-03-22]. Dostupné z: http://juniversal.org/.
12. *Sencha* [online] [cit. 2020-03-22]. Dostupné z: https://www.sencha.com/products/extjs/.
13. *Appcelerator* [online] [cit. 2020-03-22]. Dostupné z: https://www.appcelerator.com/.

Tabulka 1: Android Studio

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	0.0119	0.3512	0.0165	0.0121
Preferences Read 16B	100	0.0012	0.0029	0.0013	0.0012
Preferences Write 1K	100	0.0124	0.3652	0.0172	0.0127
Preferences Read 1K	100	0.0012	0.0029	0.0012	0.0012
File Write 16B	100	0.2822	0.5343	0.2954	0.2888
File Read 16B	100	0.0467	0.0937	0.0514	0.0503
File Write 1K	100	0.2962	0.5369	0.3123	0.3063
File Read 1K	100	0.0488	0.0922	0.0522	0.0511
Vibration	100	3.676	30.037	24.414	24.183
Audio	100	27.029	126.588	48.675	43.789
Ackermann 3/9	100	83.018	158.905	84.593	83.774
Ackermann 3/11	100	1490.283	1549.717	1513.941	1516.628

Tabulka 2: Nástroj PhoneGap

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	0.021	0.106	0.0259	0.025
Preferences Read 16B	100	0.001	0.007	0.0033	0.003
Preferences Write 1K	100	0.026	0.055	0.0294	0.028
Preferences Read 1K	100	0.001	0.006	0.0029	0.002
File Write 16B	100	24.59	55.94	41.2289	42.92
File Read 16B	100	17.38	31.24	27.6421	28.58
File Write 1K	100	23.97	55.02	41.9596	43.61
File Read 1K	100	16.09	31.81	28.2105	29.51
Vibration	100	19	53	29.93	29
Audio	100	6	56	13.55	13
Ackermann 3/9	100	159	233	160.07	159
Ackermann 3/11	100	2560	2570	2564.05	2565

Tabulka 3: Nástroj Felgo

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	9.3	15.32	12.0195	12.01
Preferences Read 16B	100	0.11	0.56	0.1412	0.11
Preferences Write 1K	100	9.54	17.66	13.6143	13.74
Preferences Read 1K	100	0.17	0.56	0.2082	0.215
File Write 16B	100	0.11	0.19	0.118	0.12
File Read 16B	100	0.08	0.14	0.0885	0.09
File Write 1K	100	0.116	0.241	0.12	0.117
File Read 1K	100	0.094	0.101	0.0977	0.098
Vibration	100	4	28	24.12	24
Audio	100	3	4	3	3
Ackermann 3/9	100	56	158	58.1	57
Ackermann 3/11					

Tabulka 4: Nástroj Ionic

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	2.774	5.086	4.7601	4.843
Preferences Read 16B	100	2.819	4.998	4.6585	4.7405
Preferences Write 1K	100	3.523	5.269	4.9911	5.0675
Preferences Read 1K	100	2.268	5.222	4.7004	4.774
File Write 16B	100	9.5	29.8	14.39	14.5
File Read 16B	100	4.2	12.5	9.338	9.8
File Write 1K	100	9.4	19.7	14.742	14.75
File Read 1K	100	3.6	13.6	9.318	9.5
Vibration	100	1	7	2.48	2
Audio					
Ackermann 3/9	100	158	275	159.68	158
Ackermann 3/11	100	2541	2546	2542.7	2543

Tabulka 5: Nástroj Xamarin

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	0.1984	0.2832	0.2042	0.2038
Preferences Read 16B	100	0.0892	0.1066	0.0911	0.0899
Preferences Write 1K	100	0.2075	0.2915	0.2122	0.2116
Preferences Read 1K	100	0.0925	0.1085	0.0952	0.0953
File Write 16B	100	0.217	0.3081	0.2234	0.2203
File Read 16B	100	0.0665	0.0801	0.0677	0.0675
File Write 1K	100	0.2227	0.3494	0.229	0.2255
File Read 1K	100	0.0805	0.0953	0.0827	0.0825
Vibration	100	11.022	39.299	32.242	32.781
Audio	100	25.131	110.992	47.428	42.602
Ackermann 3/9	100	78.489	118.617	79.312	78.873
Ackermann 3/11	100	1602.888	1660.233	1627.792	1630.412

Tabulka 6: Nástroj Codename ONE

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	0.241	0.486	0.2515	0.246
Preferences Read 16B	100	0.0001	0.001	0.0009	0.0004
Preferences Write 1K	100	0.261	0.447	0.2714	0.266
Preferences Read 1K	100	0.0001	0.001	0.0007	0.0004
File Write 16B	100	0.124	0.253	0.1297	0.127
File Read 16B	100	0.093	0.154	0.0958	0.095
File Write 1K	100	0.13	0.263	0.1366	0.133
File Read 1K	100	0.098	0.156	0.1018	0.101
Vibration	100	3	33	25.57	26
Audio	100	81	89	86.421	86
Ackermann 3/9	100	84	183	85.48	84
Ackermann 3/11					

Tabulka 7: Nástroj Expo

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	53.5	127.5	87.235	74.6
Preferences Read 16B	100	49.4	100.2	74.33	74.6
Preferences Write 1K	100	81.6	177.3	133.021	135.5
Preferences Read 1K	100	80.2	164.3	124.331	126.7
File Write 16B	100	8.308	13.572	10.8544	10.732
File Read 16B	100	7.272	10.237	8.9596	9.026
File Write 1K	100	8.553	13.018	10.4766	10.377
File Read 1K	100	8.019	9.908	9.0926	9.092
Vibration	100	1	5	1.04	1
Audio	100	75	1458	250.6	119
Ackermann 3/9	100	1201	1360	1207	1209
Ackermann 3/11					

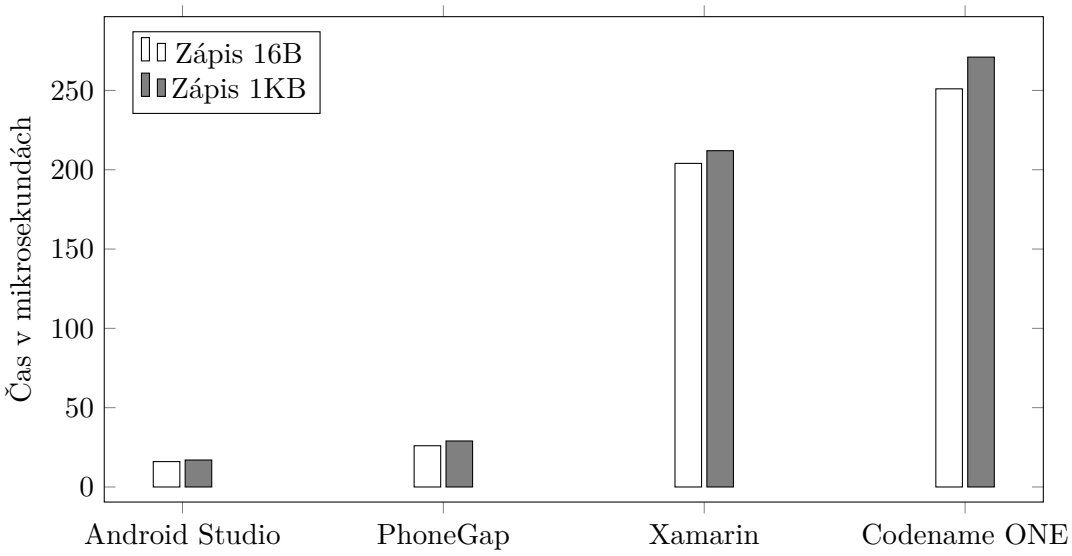
Tabulka 8: Nástroj Corona

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	20	30	22.9	20
Preferences Read 16B	100	0.02	0.02	0.02	0.02
Preferences Write 1K	100	20	30	23.5	20
Preferences Read 1K	100	0.02	0.02	0.02	0.02
File Write 16B	100	0.06	0.09	0.0736	0.07
File Read 16B	100	0.08	0.09	0.084	0.08
File Write 1K	100	0.6	1	0.8	0.8
File Read 1K	100	0.08	0.09	0.0865	0.08
Vibration	100	1	1	1	1
Audio	100	1	1	1	1
Ackermann 3/9	100	600	700	623	600
Ackermann 3/11	100	9000	10000	9000	9000

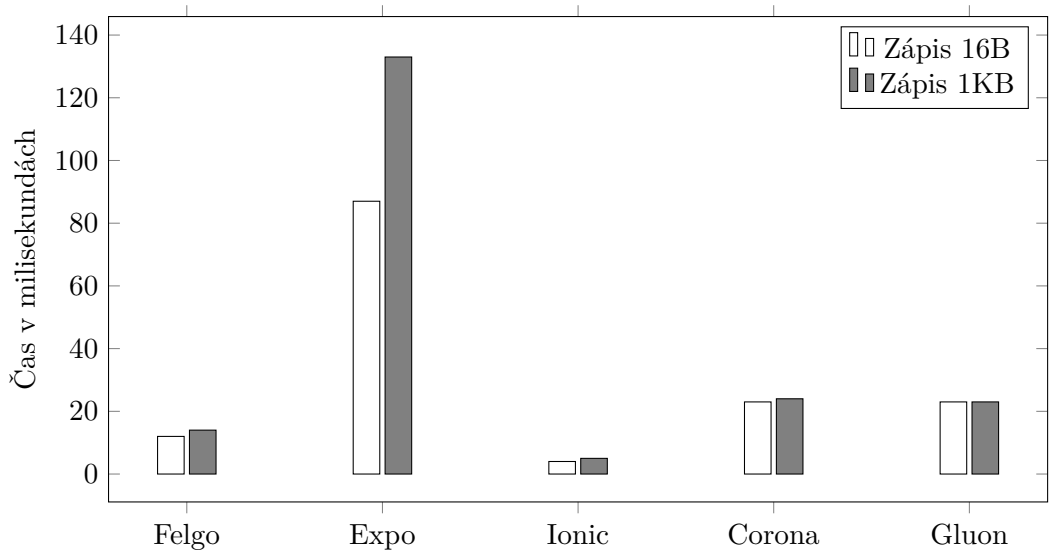
Tabulka 9: Nástroj Gluon

Test	Samples	Min	Max	Average	Median
Preferences Write 16B	100	14.598	33.724	22.978	22.327
Preferences Read 16B	100	0.0103	0.0529	0.0172	0.0174
Preferences Write 1K	100	13.313	39.168	23.005	22.023
Preferences Read 1K	100	0.0111	0.1151	0.0235	0.0186
File Write 16B	100	2.6494	4.2857	2.7974	2.7533
File Read 16B	100	0.4742	0.9663	0.5299	0.5276
File Write 1K	100	2.6418	4.3614	2.7936	2.7607
File Read 1K	100	0.4718	0.9887	0.5321	0.5311
Vibration	100	5.2020	32.165	25.81	25.862
Audio	100	64.514	68.782	66.214	65.232
Ackermann 3/9	100	83.270	111.01	84.066	83.752
Ackermann 3/11	100	1508.1	1558.8	1526.9	1527.0

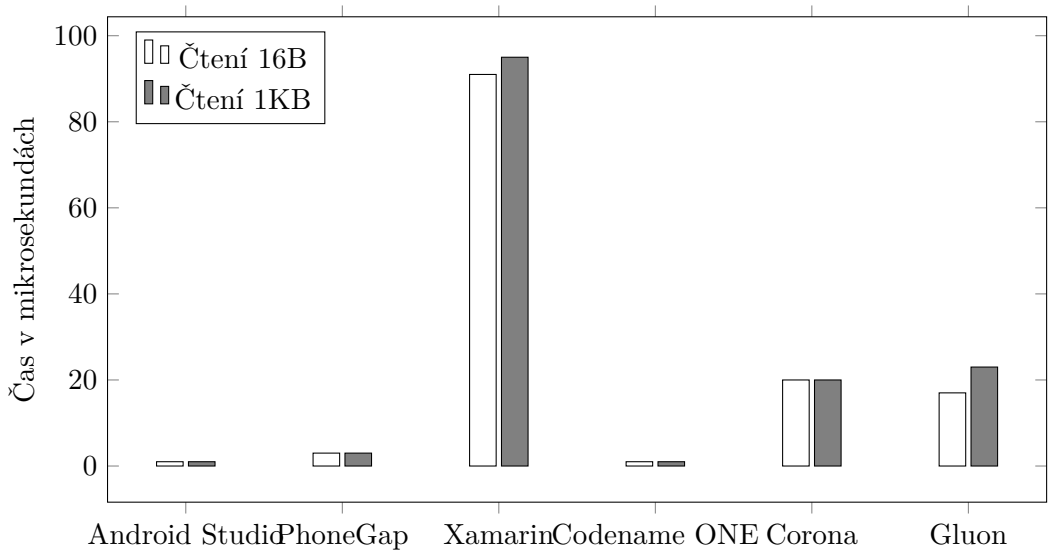
Obrázek 7: Zápis do preferencí (pod 1 ms)



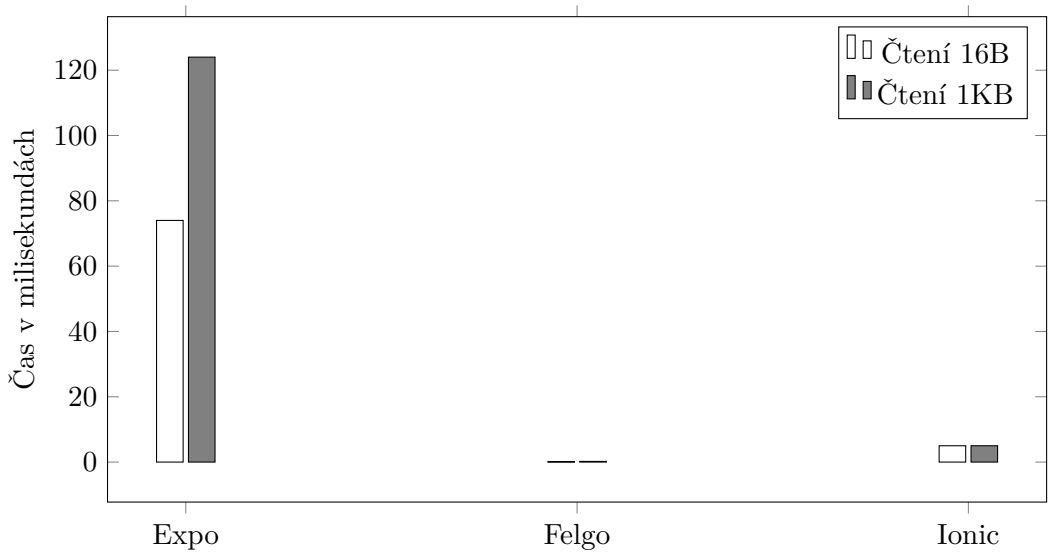
Obrázek 8: Zápis do preferencí (nad 1 ms)



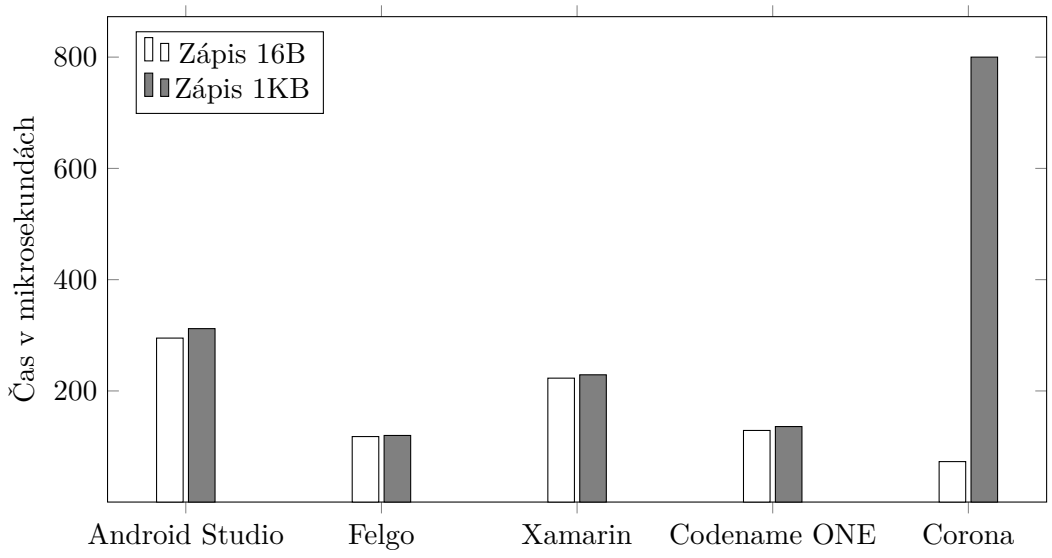
Obrázek 9: Čtení z preferencí (pod 0,1 ms)



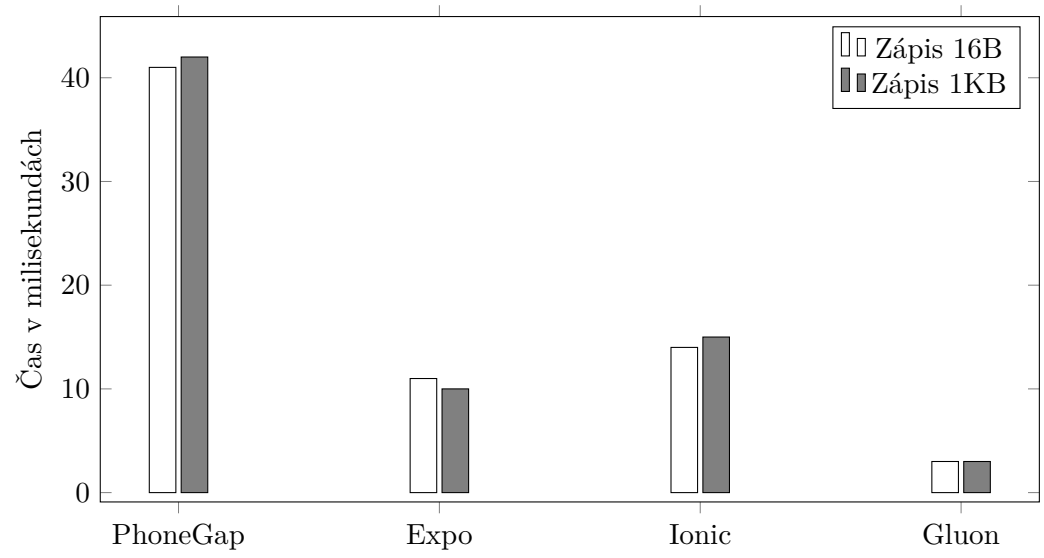
Obrázek 10: Čtení z preferencí (nad 0,1 ms)



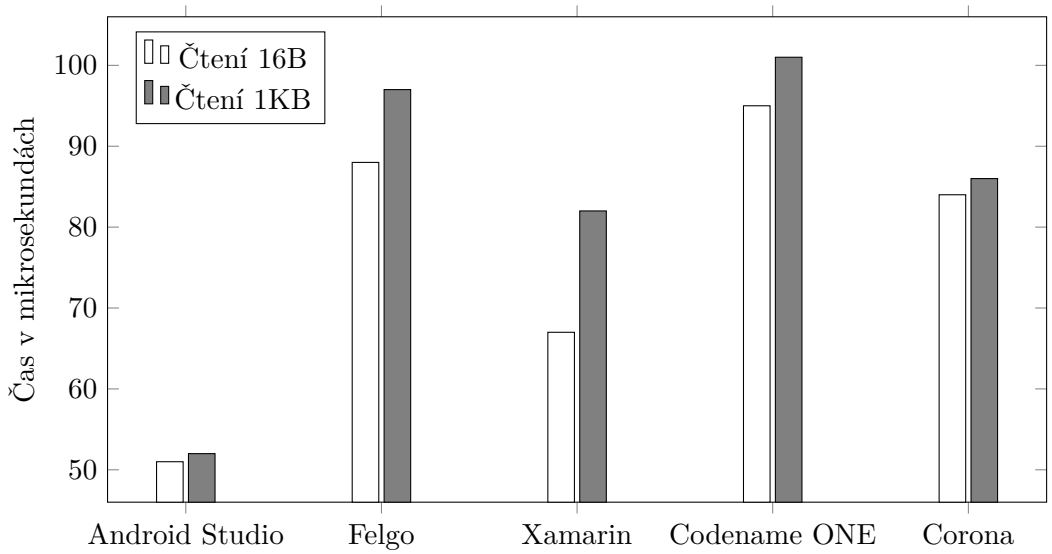
Obrázek 11: Zápis do souboru (pod 1 ms)



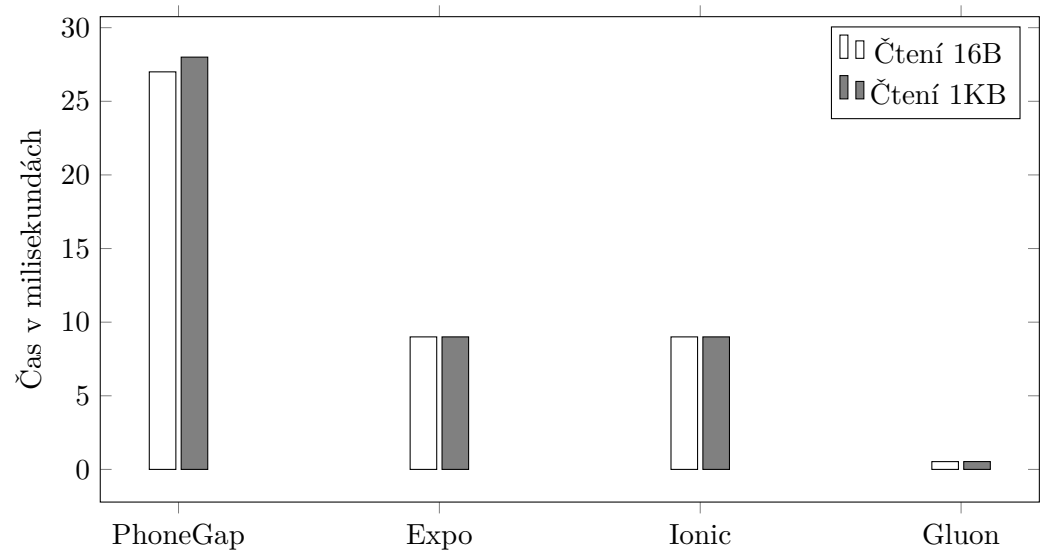
Obrázek 12: Zápis do souboru (nad 1 ms)



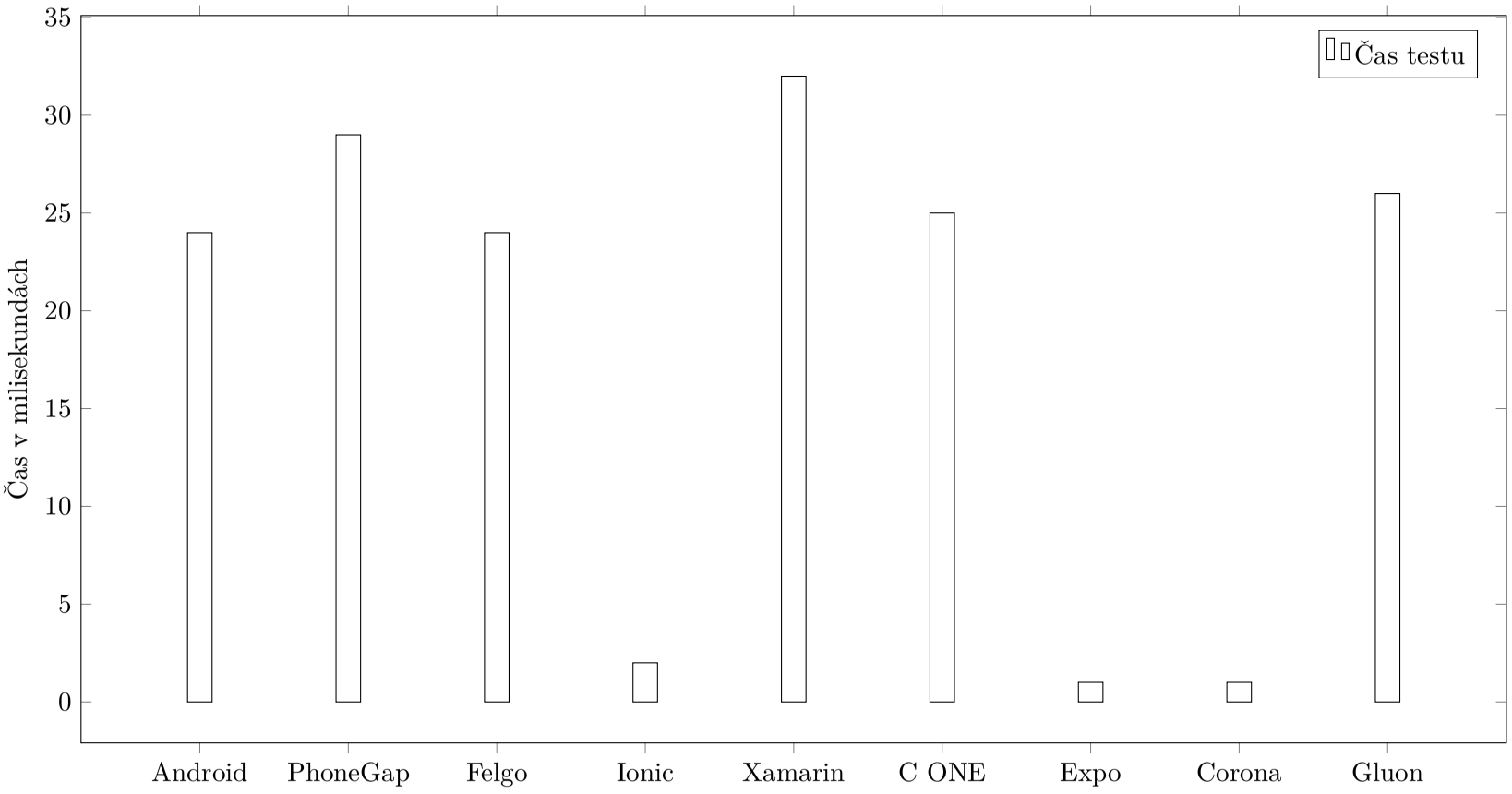
Obrázek 13: Čtení ze souboru (pod 0,1 ms)



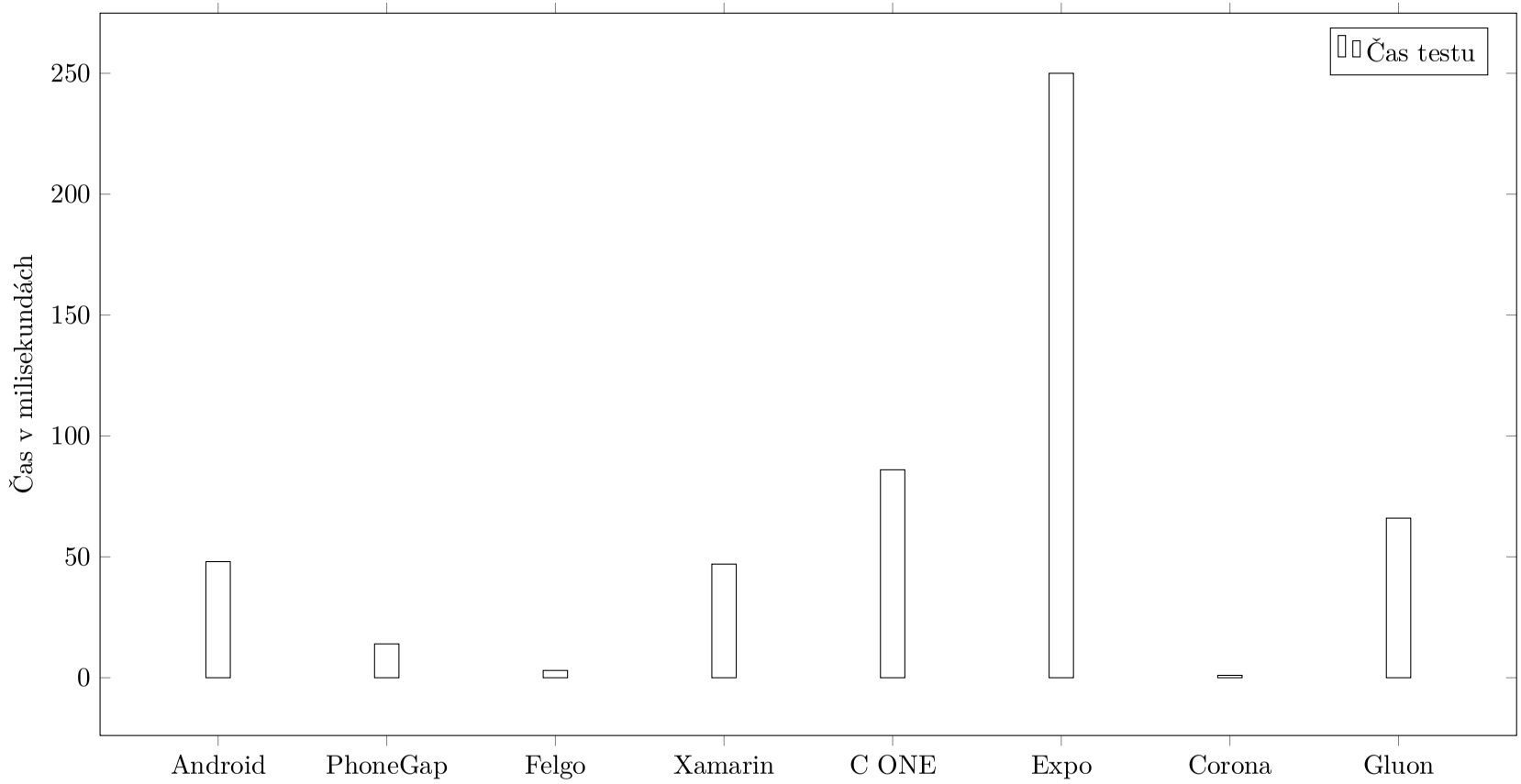
Obrázek 14: Čtení ze souboru (nad 0,1 ms)



Obrázek 15: Vibrace



Obrázek 16: Audio



Obrázek 17: Ackermann